Chapter 6

Threat-Oriented Security Model for Securing Software Systems

Software security in this digital era is encountered with new threat perceptions which were more or less unknown in early phases of software development life cycle. Proactive threat management of today demands that vigorous remedial steps may have to be taken to guard against multi-faceted threats to make the security of the system meaningful and rather fool proof. Taking broader view of threat management in the present day security environment, this chapter presents threat-oriented security model to counter both known and unforeseen threats. This model is multi-layered security architecture and has been embedded in the risk analysis segment of spiral framework to enhance and strengthen security of software systems. As the spiral process is cyclic and iterative in nature, subsequent passes of spiral process progressively reduce risk associated with threats encountered in the software development life cycle. Some in-built strategic guidelines have also been discussed in this chapter as a dual protection strategy.

6.1 Introduction

In today’s interconnected computing environment, probability that an application will encounter nefarious usage is greatly heightened (Young et al., 2010). Innovative security measures may therefore have to be adopted to detect and robustly respond to security threats encountered in this information age. However, it is not possible to prevent such threats with existing security solutions discussed in previous chapters being adopted in isolation (Hentea, 2007). We have therefore integrated these paradigms to evolve threat-oriented security model to avert both known and unforeseen threats in risk management to the extent possible. The proposed model presented in this chapter is a multi-layered security mechanism embedded in risk analysis segment of spiral process to identify and eliminate these threats progressively for risk reduction to the acceptance level of software security. Treatment of security threats in the earlier phases of system development reduces
overall development cost due to absence of variety of vulnerabilities. These security improvements also indirectly translate into operational cost savings since less time and money is wasted recovering from attacks enabled by software security vulnerabilities (Hein & Saiedian, 2009).

The proposed model is a four layered approach and is shown in Figure 6.1.

![Figure 6.1: Threat-Oriented Security Model -A Layered Architecture](image_url)

Identification of threats is the function of first layer. Here threat modeling process has been used for identification of known threats whereas a detection mechanism has been planted using research honeytokens in conjunction with statistical model for detection of unforeseen threats. Threat modeling provides a structured way to secure software design by allowing security designers to accurately estimate the attacker’s capabilities in respect of known threats. Research honeytokens have been adopted here to capture new or unpredictable threats in the field of software security. Honeytokens are strongly tied to honeypots and are defined as information system resources whose value lies in unauthorized or illicit use of resources (Spitzner, 2002; 2003). In the proposed work data
fusion and information correlation from two sources i.e. statistical model and research honeytokens ensures that possible unforeseen threats which went unnoticed previously are also discovered.

Threats identified above are then optimized for mitigation using Layered Threat Elimination Model (LTEM) and Optimal Countermeasures Identification Method (OCIM) in the second layer. LTEM gives optimum number of threats needing mitigation in the light of security policy of the system, classification of threats, prioritization of threats using E-DREAD algorithm and cost-benefit analysis for effective and economical risk management. Thereafter, OCIM approach discussed in chapter 5 is applied which gives minimum number of attack nodes for mitigation of above identified threats obtained as output of LTEM making it all the more optimal. Appropriate countermeasures identified with the help of hybrid technique are then applied on these optimum attack nodes in the third layer of proposed model using multi-agent system planning. Multi-agent system planning involves multiple agents, each performing a sub-task of a complex problem, and the solution to it is obtained by merging the sub-solutions provided by different agents. In MASPTA, these agents work in unison to avoid threats to a software system with their predetermined action plans executed by individual agents as per the schedule generated as discussed in detail in Chapter 4.

Evaluation of risk reduction in respect of both known and unforeseen threats after adopting necessary countermeasures is the responsibility of meta-agents inducted in the fourth layer along with fuzzy logic. Here, a monitoring meta-agent has been introduced to monitor performance of active agents employed in layer above and also to monitor variations in security goals which form input for fuzzy inference system. Another meta-agent i.e. response agent in this framework acts autonomously, reason and takes decisions to save the system from being compromised in the light of variations in the security level generated as output of fuzzy inference system (Gandotra et al., 2011a).

This four-layered security model is then placed in the risk analysis segment of spiral framework to enhance and strengthen security as a part of proactive risk management. As the spiral process is cyclic and iterative in nature, subsequent passes of spiral process progressively reduce the risk associated with threats encountered in the
development life cycle. This innovative approach helps designers and developers to anticipate both known and unforeseen threats to a system under development, and so also the type and mode of realization of such threats. Thereafter, appropriate countermeasures are applied to avert these threats during the design phase itself to reduce design level vulnerabilities which are a major source of software risks. However this model may have to be revisited periodically so that it stays current to meet new evolving threat perceptions (Gandotra et al., 2012; Bedi et al., 2012a).

This chapter has been categorized into five sections. Section 6.2 discusses research carried out by different researchers and authors in this area. Section 6.3 describes threat-oriented security model for avoidance of both known and unforeseen threats. Adoption of proposed model in the spiral framework has been given in Section 6.4. Section 6.5 makes comparative evaluation of proposed model with traditional security solutions.

6.2 Literature Review

In this digital era, risk management is facing multi-pronged attacks from organized hackers. As can be seen in previous chapters different authors and researchers (Swiderski & Snyder, 2004; Howard & Le Blanc, 2003; McGraw, 2006; Davis, 2005; Hein & Saiedian, 2009) have worked on different aspects of threat management as a part of secure software engineering, but we have not found any comprehensive approach in literature. We have therefore augmented efforts made by leading researchers in this area by integrating different innovative techniques to evolve a threat-oriented security framework as a defense strategy for securing software systems against both known and unforeseen threats. This approach is different from traditional security solutions as it averts threats to software systems optimally using multi-agent system planning.

Moradian and Hakansson (2008; 2010) have introduced the approach to solving security problems using meta-agents in multi-agent system. In threat-oriented security framework presented in this chapter, meta-agents in conjunction with fuzzy logic have been inducted for monitoring and management of threats in multi-agent environment (Gandotra et al., 2011a). Spiral model developed by Boehm (1988) based on a risk-driven and cyclic approach has also contributed towards risk reduction in the proposed work for
developing secure software systems. Honeypots, a new technology with enormous potential is used these days for detecting and gathering information about security threats. Lance Spitzner (2002; 2003) has given a new dimension to this area of research and has formalized the idea of honeytokens. Different authors (Barfar & Mohammadi, 2007; White, 2010) have also worked in this field to further strengthen security of software systems by developing responses to new attacks of black hats. We have also adopted the above concept in threat oriented security model discussed in this chapter to capture unforeseen threats for mitigation (Gandotra et al., 2012, Bedi et al., 2012a).

Howard et al. (2003) have given the concept of Fail to Secure Mode as one of the secure design principles. This concept has been adopted as a proactive step in threat management to save the system from being compromised even if the attacker intrudes into the system. In this chapter, we have proposed some relevant steps and strategies to enhance and strengthen this design principle so that confidential data can not be tampered in any case.

Cross threats from external insiders these days are proving more challenging for threat management due to changing business perceptions. Franqueira et al. (2010) has pointed out some security challenges that may emanate from external insiders in their paper. This class of threats is much more damaging due to multiple ownership and no system can be fully secure without adopting effective remedial measures against these threats. In our approach Cross Threats Evaluation and Control (CTEC), we have tried to enhance the existing security measures to cover threats from external insiders to the extent possible (Gandotra et al., 2011c).

### 6.3 Threat-Oriented Security Model

Traditional technologies used these days can prevent known threats but they are unable to identify and avert unforeseen threats. These unforeseen threats may be equally instrumental in compromising security of the system and also require proactive security measures to avert them. Threat-oriented security model is a step in this direction and is embedded in risk analysis segment of spiral process to identify and eliminate both known and unforeseen threats progressively for risk reduction to the acceptance level of software
security. The proposed model has been subdivided into four layers and has been evolved by integrating various paradigms to match the evolutionary nature of threats manifested these days as shown in Figure 6.2. This model overcomes the deficiency of shallowness in a single-layer defense strategy and provides multi-level security cover to avert threats.

Figure 6.2: Expanded View of layers of Threat-Oriented Security Model

In the above framework every successive layer takes the output of previous layer as input and tries to enhance security for averting threats to software systems. The probability of realization of different threats decreases with increase in number of security layers and this concept has been suitably applied by incorporating innovative security mechanisms discussed in previous chapters as a part of proactive threat management. However the number of layers depends on the criticality of the application, financial constraints and the security requirements of the system. The developers can adopt one or more layers of the proposed model during design and development stage of SDLC. As the proposed architecture is incremental in nature more layers can be added if so required.
meeting future security requirements also. Different layers of this proposed security model are explained below.

### 6.3.1 Layer 1: Identification of Known and Unforeseen Threats

This layer presents a new integrated security mechanism to identify both known and unpredictable threats. Known or predictable threats have been identified using threat modeling process while research honeytokens in conjunction with statistical technique has been adopted for identification of unforeseen threats as shown in Figure 6.3 below.

![Figure 6.3: Integrated Mechanism for identification of Known and Unseen Threats](image_url)

Threat modeling process as discussed in detail in Chapter 2 and 3 provides a structured way to secure software design which involves understanding an adversary’s goal in attacking a system based on system’s assets of interest. Threat modeling process consists of characterizing the security of the system, identifying assets and access points and determining known threats. Although above mechanism provides broad view of identification of known threats but security has no meaning unless unforeseen threats to the system are not identified and taken care of in the security framework of proactive threat management. These unforeseen threats may be due to exploitation of vulnerabilities left unnoticed during design phase or adaptive threats by sophisticated hackers. The strategy for capturing these unforeseen threats to the system adopted in this proposed model is given below.

Research honeytokens are a new concept to the computer security arena which in fact is most interesting implementation of a honeypot to detect unforeseen threats. The term
honeytoken was first coined by Augusto Paes de Barros in 2003 in a discussion on honeypots. Honeytokens are honeypots that are not physical devices but are a digital entity that would look attractive and useful to an attacker (Spitzner, 2003). This concept has been further augmented by Lance Spitzner to proactively gather information about security threats by providing a real system with real applications to adopt remedial measures against them. A research honeytoken is a highly flexible tool with no production value and any interaction with honeytoken means malicious or fraudulent activity. Research honeytokens deployed should be advanced one and updated regularly so that they are capable of attracting more attackers without them knowing it (Spitzner, 2003; White, 2010).

In the proposed model, research honeytokens have been used as a digital entity planted during the design phase of the software life cycle itself to procure information about attacker profiles in respect of unforeseen threats. Two research honeytokens in this work have been deployed to trap malicious users for any illegal activity which has been demonstrated experimentally in the case study given in next chapter. These research honeytokens are then closely monitored by system administrator to identify new or unseen vulnerabilities which are being exploited by the attacker to compromise security of the system. Forensic analysis of collected data from these honeytokens gives preferred attack patterns against unforeseen threats.

While research honeytokens represent a powerful tool to identify unforeseen threats but they have their own limitations. There may be cases when the attacker may not interact with them and may have a little value as observable. Moreover attackers can also use a research honeytoken maliciously to attack other non-honeytoken systems. This deficiency has been overcome to some extent by adopting statistical technique with honeytokens in the proposed model. Statistical based systems use statistical models to detect malicious activity. These systems adapt to different system behaviors or occurrences and try to develop a usage pattern. Predefined variables monitored over a time period are calculated for a test value. If this value is above the user defined threshold, then an alert is triggered. This approach does not require any predefined attack patterns and is capable of detecting new or unknown attacks (Yongro, 2005). Fusion and information correlation of data received from these two sources provides much better detection ability for wide range of threats and attacks at an early stage. Moreover, the normalcy depends on
correlations among different parameters. The independent values of two different parameters determined using research honeytokens and statistical model can be taken normal, but their combination can show abnormality (Debar & Wespi, 2001; Gupta, 2007). In this way correlating information from multiple sources helps in detecting many potential unpredictable attacks which can result in security compromise of a system.

Layer-1 thus provides a means of determining both known and unforeseen threats that serve as input to Layer-2 for determining optimum number of threats to be mitigated.

### 6.3.2 Layer 2: Optimization of Threats

Known and unforeseen threats identified above are then optimized for mitigation using LTEM and OCIM techniques in this layer. LTEM as discussed in Chapter 4 gives threats which are not that relevant to the security of the system and can be safely ignored. It gives optimum number of threats needing mitigation in the light of security policy of the system, classification of threats, prioritization of threats using E-DREAD algorithm and cost-benefit analysis for effective and economical risk management. Thereafter, OCIM approach as described in Chapter 5 is applied which gives minimum number of attack nodes for mitigation of above identified threats obtained as output of LTEM making it all the more optimal.

Thus application of these two techniques in this layer helps in reducing considerably number of countermeasures to be applied to save the system from being compromised in an optimal way. However this degree of reduction is not constant and may vary from application to application and the environment in which it works.

### 6.3.3 Layer 3: Mitigation of Threats

In this layer, countermeasures are applied on optimum attack nodes identified in the previous layer of proposed model using multi-agent system planning. These countermeasures are identified with the help of hybrid process diagram which is a manifestation of various stages starting from threat to mitigation level as discussed in Chapter 3. The countermeasures so identified for security of software system are assigned as action plans to multiple agents. These agents deployed on attack nodes executes their
predetermined action plans to avert above identified threats optimally as described in
Chapter 4. Experimental results corresponding to implementation of above agents have
been shown in case study presented in next chapter to prove its feasibility.

6.3.4 Layer 4: Monitoring and Management of Threats

Layer-3 is able to avoid threats optimally from attackers using multi-agent system
planning. Adoption of this defensive mechanism brings our system in either of two states
as detailed below.

**Safe State:** In this state, security measures are sufficient to avoid threat to keep the system
in Safe mode.

**Failed State:** In this state, security measures have failed to stop the execution of threat
bringing the system in failed mode.

Thus we see that the present system works on binary conditions of either success
or failure of the security mechanisms. There are no means or mechanism at present to
protect the systems from complete failure which is unacceptable in this information age.
This failure can result in catastrophic consequences in terms of leakage of confidential
data so vital to users. We have supplemented the measures taken above by induction of
fuzzy logic as a step further towards secure software systems. This has helped us in
removing the above brittleness by evolving an intermediate stage i.e. Partially Secure State
(PS)/Yellow Zone between Fully Secure State (FS) / Green Zone and Failed State (FS) /
Red Zone as discussed in Chapter 3. In this chapter, this concept has been usefully applied
in evolving a defense strategy for monitoring and managing threats in secure software
engineering as shown in Figure 6.4.

As we know, major goals of any secure software system are preservation of security
goals i.e. Confidentiality (C), Integrity (I) and Availability (A) of the system. Once
countermeasures are applied, it is presumed that the system is in the safe state and is able to
maintain these security goals. When some threat starts taking place the values for security
goals also start coming down and correspondingly change the output security level. This
concept has been effectively used in this layer to evolve a defense mechanism to neutralize
present day multifaceted security threats as shown in Figure 6.4. This enhancement has been made possible by using meta-agents in multi-agent system in conjunction with fuzzy logic. Meta-agents which are adaptive and reactive in nature are used as management, coordination, matching and checking agents. The main role of meta-agents is to shift between strategies and make decisions. These strategies can be implemented in a single agent or in a group of agents (Moradian & Hakonsson, 2008; 2010). In this mechanism, meta-agents communicate with software agents as well as other meta-agents to achieve the goal of secure software system. Here, meta-agents have the responsibility of monitoring and controlling other software agents deployed in previous layer as well as evaluating the security level of the system to take decisions for appropriate additional security measures so that the attacker may not succeed in his mission. The framework of this defense mechanism has been subdivided into three units as shown in the diagram given below:

![Figure 6.4: Defense Mechanism for Monitoring and Managing Threats](image)

In Unit-I, a monitoring meta-agent has been inducted to monitor continuously all the active agents employed in Layer-3 and their performance regarding execution of their action plans. This agent also constantly monitors variations in security goals and passes them on as input to the fuzzy inference system (FIS) in Unit-II which in turn generates overall security level of the system. The generated security level by FIS in Unit-II is taken care of by other meta-agents in Unit-III. These meta-agents monitor variations in the security level and analyze to determine the state of security of the system for taking actions necessary to save the system from being compromised.

**Secure State / Green Zone**

If the analysis of security level indicates that the system is in green zone, no further action is be taken by the meta-agents in Unit-III. This means that either there is no threat to the system during observation or the countermeasures adopted at Layer-3 are able to avert the threat.
Partial Secure State / Yellow Zone

In this state the attacker has succeeded in intruding the system with partial success in achieving his goal of compromising the system. This may be due to the fact that the countermeasures applied at Layer-3 are not able to prevent the threat completely or some unknown threat has taken place resulting in partial damage to the system. This state is in fact an ‘Alert Signal’ to meta-agents who in response take appropriate additional measures to avoid the system from entering into the Failed State / Red Zone. These additional measures may include induction of complex algorithms, biometric features, multifactor security mechanism or other features that may be necessary to prevent the attacker from achieving his goal. Here threat catalogue can also be a useful tool for meta-agents as the data stored therein will be helpful in achieving the objective of maintaining secure state in such cases (Oladimeji et al., 2006).

Failed State / Red Zone

Even after adoption of additional countermeasures if meta-agents find that security level is moving towards Failed State / Red Zone, they initiates prompt remedial measures. This state may be due to failure of the countermeasures applied earlier to avert the threat; or this situation may also arise when there is unknown threat to the system which needs to be tackled at a different level. In this case the attacker is able to compromise with the security of the system by penetrating all system defenses and become part of the trusted system itself. This is a dangerous situation indeed and has to be addressed even if meta-agents have to shut down the system completely or partially for some time. This reduces the propagation of attack impact and guard the system against possible damage which would have occurred otherwise. Here even if the attacker breaches the security of the system, he will not be in a position to exploit or damage the system completely in any case which is also termed as “Fail to Secure Mode”.

Thus we see that use of fuzzy logic in conjunction with meta-agents in this proactive mechanism is revolutionary in nature and is capable of meeting threats posed by present day security environment. Experimental results in this respect have been shown in case study in Chapter 7 to validate its effectiveness.
6.3.5 Inbuilt Security Guidelines in Threat Management

Till now we have discussed proactive security measures adopted at different layers of proposed threat-oriented security model to avert present day threats to the extent possible. However modern day hackers not only make it possible to get to defensive techniques employed by defenders but are coming up with new threat perceptions compound in nature with changing attack modes also. This has necessitated a comprehensive security mechanism which can take up these new challenges effectively. The guidelines suggested in this chapter are in-built and can be applied judiciously wherever required to strengthen the security model described above. In case remedial steps taken above fail to save the system from being compromised, the ‘Fail to a Secure Mode’ saves the system security even if intruder crosses the system boundary or the outer security cover. ‘Fail to a Secure Mode’ means that the application has not disclosed any data that would not be disclosed ordinarily or the data still cannot be tampered with and so on (Howard & LeBlanc, 2003). The application of this in-built proactive step is an enhancement of traditional defense mechanisms where penetration into periphery defenses means that the system has been compromised. This augmentation is a two-fold protection strategy as against current state of art of defense technology. Some of the important features to be adopted to achieve ‘Fail to a Secure Mode’ are given below.

- The system to be protected is first analyzed for sensitive and critical assets which need protection to save the system from being fully compromised. The sensitive assets so identified are placed in different containers so that the attacker will not be able to tamper all of them at the same time. Even if a container is compromised other containers will continue to work as the exposed attack surface has been narrowed. In the mean time response mechanism can isolate the affected container and localize the attack’s impact. This step helps us to achieve the objective of ‘Fail to a Secure Mode’ as the intrusion surface has been localized and isolated. It is advisable to place the containers holding sensitive and critical data at random and unconventional places. This serves as camouflage making the detection of secret data difficult for access by the attacker even if he intrudes into the system boundary.

- Encryption of sensitive data may be resorted to wherever required so that intruders may not be able to use data for malicious purpose. Cryptographers have come out
with various new encryption schemes like Predicate Encryption, Identity–Based Encryption, Attribute–Based Encryption etc. (Goyal et al., 2006; Katz et al., 2008). Some designers are also using more complex codes these days for information hiding through obscurity like obfuscation techniques to meet above requirements (Witkoska, 2006). Use of these techniques can go a long way to enhance security of the system to maintain ‘Fail to a Secure Mode’.

• “Least Use Privileges” and “Separation of Privileges” are the two important design principles which help in reducing the attack surface so that access to critical assets does not pass onto malicious users. Various authentication schemes provide protection to avoid its use by unauthorized users. In this case even if malicious user intrudes into the system he can not have access due to authentication schemes implied to avert such situations e.g. password, private key and use of biometric authentication and so on. This automatically brings the system’s security to ‘Fail to a Secure Mode’.

• Defense-in-depth strategy overcomes the deficiency of shallowness in a single-layered defense strategy and provides multi-level security cover to avert threats. In this case even if the intruder gets into the system he will not have access to assets due to multi-layered defense mechanism.

• In addition, a new category of threats known as cross threats has arisen these days due to changing business perceptions as more and more organizations are invariably resorting to outsourcing, partnerships, joint ventures for business promotion and financial gains. These threats are coming from a set of people having characteristics of both outsiders and insiders and hence termed as external insiders. This class of threats is proving more challenging for risk management due to multiple ownership. Gandotra et al. (2011c) have proposed some proactive in-built security measures to be adopted to avoid such cross threats to save the system from malicious users in their research work as a part of threat management.

Appropriate security features can be incorporated during the design phase of software development life cycle in the light of guidelines given above as and when required or considered necessary for securing software systems.
6.4 Threat-Oriented Security Model in Spiral Framework

The proposed threat-oriented security model is now inducted in risk analysis segment of the spiral framework as shown in Figure 6.5 below.

Figure 6.5: Induction of Threat-Oriented Security Model in Spiral Framework

Spiral model is an evolutionary version of incremental prototyping developed by Boehm in 1988 (Boehm, 1988). This software process model is based on a risk-driven and cyclic approach and provides the potential for rapid development of increasingly more complete versions of the software while decreasing its degree of risk. A spiral model is a realistic approach to the development of large-scale systems and is divided into set of framework activities as shown in the diagram. As this evolutionary process begins, the software team performs activities that are implied by a circuit around the spiral in a clockwise direction, beginning at the centre and risk is considered as each revolution is made. Because the software evolves as the process progresses, the developer and customer better understand and react to risks at each evolutionary level to meet the present day
security requirements of the system to be developed reducing risks before they become problematic (Boehm, 1988; Pressman, 2005).

The first circuit around the spiral has been used for identification of known threats and adoption of necessary countermeasures to mitigate them. Honeytokens in conjunction with statistical model have also been planted to detect unforeseen threats and to monitor attacker’s profile. In addition this mechanism also helps if safeguards already deployed to mitigate known threats are adequate. For instance if a detection mechanism is attacked, it proves that an attacker found a way to this mechanism. With the knowledge of these attacks, it is easier for the designers to determine and close security holes, as well as adopt additional safeguards to close the exploits to avoid the real attacks (Barfar & Mohammadi, 2007). This process is then carried out to the next segments of the spiral process for development of the software prototype. This prototype is then deployed for feedback in respect of following security parameters:

- Detection of new or unforeseen threats which may be due to exploitation of vulnerabilities that have gone unnoticed during the first iteration with the help of planted detection mechanism.
- Performance of the countermeasures applied against identified threats using meta-agents along with fuzzy inference system in multi-agent environment.
- New security features required to be adopted to cater to additional security requirements given by the customer after delivery of prototype.

On the basis of above feedback necessary security features are incorporated by the designers to remove observed anomalies in the subsequent passes around the spiral process to counter all types of threats both known and unforeseen. In this way every reiteration results in reduction of vulnerabilities at the design level and progressive risk reduction till it reaches the acceptance level to the satisfaction of consumer.

### 6.5 Proposed Model Vs Traditional Security Solutions

The proposed threat-oriented security framework is primarily meant to enhance and strengthen the security of software systems to meet present day multi-pronged threats from
organized hackers. Every layer of this model contributes towards enhancement of security measures deployed at present. This has been evolved by fusion of various existing and innovative techniques to address threats encountered these days to the extent possible. Moreover, this model has been inducted in risk analysis segment of spiral model having iterative framework providing progressive and rapid development of increasingly more complete versions of the engineered system with decreasing risk.

A comparative study of the advantages of this model over the traditional security mechanisms is as follows:

- First layer of this model captures both known and unforeseen threats as a part of proactive threat management whereas in the existing security mechanisms only known threats are taken care of.

- Second layer of the proposed model provides optimal solution for threat avoidance resulting in substantial decrease in security expenditure as compared to traditional security mechanisms which give only non-optimal/suboptimal solutions for averting threats. This optimization shows an order of magnitude improvement over classical solutions in secure software engineering.

- Induction of agents for mitigation of threats and use of meta-agents in conjunction with fuzzy logic for monitoring and management of these threats in multi-agent environment (Layer 3 and 4) makes this innovative security model unique for securing software systems. Innovative techniques adopted in these layers provide a guarded approach in respect of security whereas existing security mechanisms based on binary principle do not address this issue as such.

- In-built security guidelines proposed in this model help provide dual security protection in case the intruder crosses the system boundary or outer security cover. This dual protection has significant advantages over contemporary methods for software security.

- Integration of proposed model in the security segment of spiral framework has given a new dimension to development of secure software systems.
• Threat Mitigation, Monitoring and Management Plan (TMMP) in the proposed model is adaptive and go on updating with every subsequent round of spiral process to the point of customer’s satisfaction.

Although this integrated and comprehensive approach has many advantages over the traditional security solutions due to adoption of various innovative techniques, it has some limitations also.

• Since this model is embedded in the spiral framework, it is ideal for the development of large-scale systems and software.

• Any security hole left uncovered exploited by malicious users can be problematic.